Reading data

```
shark <- read.csv("sharktank.csv")
```

Column Names of the data

```
names(shark)
```

```
##  [1] "deal"                   "description"
##  [3] "episode"                "category"
##  [5] "entrepreneurs"          "location"
##  [7] "website"                "askedFor"
##  [9] "exchangeForStake"       "valuation"
## [11] "season"                 "shark1"
## [13] "shark2"                 "shark3"
## [15] "shark4"                 "shark5"
## [17] "title"                  "episode.season"
## [19] "Multiple.Entreprenuers"
```

Tabulating the target variable

```
table(shark$deal)
```

```
##
## FALSE  TRUE
##   244   251
```

Loding the required libraries

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.4.4
```

```
## Loading required package: NLP
```

```
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 3.4.4
```

```
## Loading required package: RColorBrewer
```

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

Creating the corpus

```
corpus <- Corpus(VectorSource(shark$description))
```

Word clound before any pre processing

```
wordcloud(corpus, max.words = 50, colors=rainbow(7))
```

The top words are and, the, that and for which does not help in a model understanding the true meaning of the textual content. So we do data cleaning or some pre processing

Data cleaning

```
corpus <- tm_map(corpus, tolower)

## Warning in tm_map.SimpleCorpus(corpus, tolower): transformation drops
## documents

corpus <- tm_map(corpus, removePunctuation)

## Warning in tm_map.SimpleCorpus(corpus, removePunctuation): transformation
## drops documents

stopwords("english")[1:10]

##  [1] "i"         "me"        "my"        "myself"    "we"
##  [6] "our"       "ours"      "ourselves" "you"       "your"
```

```
corpus <- tm_map(corpus, removeWords, c(stopwords('english')))

## Warning in tm_map.SimpleCorpus(corpus, removeWords,
## c(stopwords("english"))): transformation drops documents

corpus <- tm_map(corpus, stemDocument)

## Warning in tm_map.SimpleCorpus(corpus, stemDocument): transformation drops
## documents
```

Word clound after pre processing

```
wordcloud(corpus, max.words = 50, colors=rainbow(7))

## Warning in wordcloud(corpus, max.words = 50, colors = rainbow(7)): made
## could not be fit on page. It will not be plotted.
```

Now the top workds are Can, Make, Made, Use, Design, Product, etc. which seems to be revealing the true meaning of the textual content

Creating the document term matrix

```
frequencies = DocumentTermMatrix(corpus)
```

20 Low frequency words

```
findFreqTerms(frequencies, lowfreq = 20)
```

```
##  [1] "devic"     "new"       "children"  "easi"      "make"
##  [6] "one"       "play"      "provid"    "servic"    "offer"
## [11] "design"    "flavor"    "food"      "includ"    "line"
## [16] "product"   "store"     "cloth"     "look"      "mix"
## [21] "get"       "work"      "made"      "help"      "fit"
## [26] "fun"       "keep"      "kid"       "compani"   "also"
## [31] "can"       "like"      "protect"   "time"      "use"
## [36] "sell"      "accessori" "onlin"     "user"      "bar"
## [41] "just"      "natur"     "safe"      "custom"    "bottl"
## [46] "featur"    "allow"     "need"      "way"       "busi"
## [51] "even"      "take"      "creat"     "without"   "come"
## [56] "system"    "home"      "water"     "peopl"
```

Number of columns

```
length(names(as.data.frame(as.matrix(frequencies))))
```

```
## [1] 3501
```

Handking sparse matrix removing words that are having less than 0.5% filled with 1

```
sparse <- removeSparseTerms(frequencies, 0.997)
sharkSparse <- as.data.frame(as.matrix(sparse))
colnames(sharkSparse) <- make.names(colnames(sharkSparse))
```

Number of columns after removing sparse words

```
length(names(sharkSparse))
```

```
## [1] 1412
```

Adding the target variable to the dataset

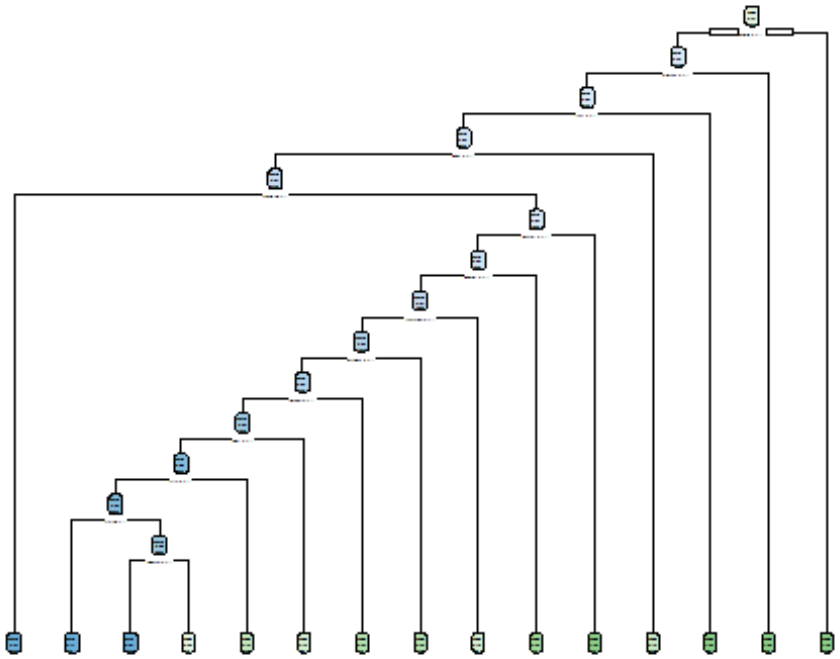
```
sharkSparse$deal <- as.factor(shark$deal)
```

Building CART model on othe sparse matrix with 900 column

```
sharkCART <- rpart(deal ~ ., data=sharkSparse, method="class")
```

Plot of CART

```
rpart.plot(sharkCART)
```



Infering CART Model

Top five words that splits the tree are

1. shape - For proposals having density of “shape” less than 0.5 does not make the deal 98% of the time
2. weight - For proposals having density of “weight” less than 0.5 does not make the deal 97% of the time
3. roll - For proposals having density of “roll” less than 0.5 does not make the deal 95% of the time
4. easi - For proposals having density of “easi” less than 0.5 does not make the deal 91% of the time
5. made - For proposals having density of made greater than 0.5 does not make the deal 79% of the time

A good proposal should contain the words “shape”, “weight”, “roll” and “easi”; and it should not contain the word “made”

Confusion matrix

```
sharkCARTPred <- predict(sharkCART, data=sharkSparse)
sharkCARTCM <- table("Actual" = sharkSparse$deal, "Prediction" = sharkCARTPre
d[,2] > 0.5)
sharkCARTCM
```

```
##        Prediction
## Actual  FALSE TRUE
##   FALSE   213   31
##   TRUE    139  112
```

Understanding the model accurady

```
accuracyCART <- (sharkCARTCM[1]+sharkCARTCM[4])/sum(sharkCARTCM)
round(accuracyCART * 100, 2)
```

```
## [1] 65.66
```

The model accuracy of CART with just the description is 65.66%

Building Random Forest with 25 tree limit

```
sharkRF <- randomForest(deal~., data = sharkSparse, ntree=25)
sharkRFCM <- table("Actual" = sharkSparse$deal, "Prediction" = sharkRF$predic
ted)
```

Confusion Matrix

```
sharkRFCM
```

```
##        Prediction
## Actual  FALSE TRUE
##   FALSE   152   92
##   TRUE    131  120
```

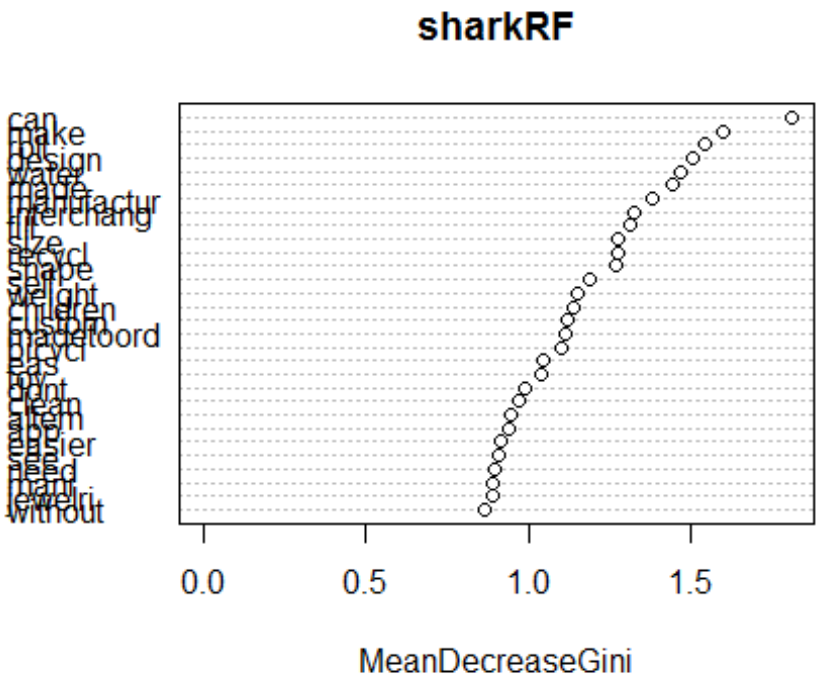
```
accuracyRF <- (sharkRFCM[1]+sharkRFCM[4])/sum(sharkRFCM)
round(accuracyRF * 100, 2)
```

```
## [1] 54.95
```

The model accuracy of RF with just the description is 53.33% which is very close to random guess of 50%

Visualizing random forest

```
varImpPlot(sharkRF)
```



Inference

Top 5 words that help classify are

1. make - This seems to be way influential than the others. Not able to infer whether it a positive impact or negative from this visualization
2. roll
3. made
4. design
5. children

Building Logistic regression model

Before building logistic regression we need to reduce the number of columns otherwise we will end up with p>n problem which will need refined logistic regression methods.

To reduce the number of columns instead of removing sparcity with 0.5% density, we will remove words that are having less than 2.05% density.

```
sparse <- removeSparseTerms(frequencies, 0.9795)
sharkSparseGlm <- as.data.frame(as.matrix(sparse))
```

```r
colnames(sharkSparseGlm) <- make.names(colnames(sharkSparseGlm))
length(names(sharkSparseGlm))
```

```
## [1] 133
```

The column count now is reduced to 133 compared to 1400

Adding the target variable to the dataset

```r
sharkSparseGlm$deal <- as.factor(shark$deal)
```

Logistic regression model

```r
sharkGlm <- glm(deal~., data = sharkSparseGlm, family = "binomial")
sharkGlmCM <- table("Actual" = sharkSparseGlm$deal, "Prediction" = sharkGlm$f
itted.values > 0.5)
```

Confusion Matrix

```r
sharkGlmCM
```

```
##        Prediction
## Actual  FALSE TRUE
##   FALSE   178   66
##   TRUE     76  175
```

McFaden R2 to evaluate the model

```r
round((1 - (sharkGlm$deviance/sharkGlm$null.deviance))*100, 2)
```

```
## [1] 26.6
```

The model explains about 27% of variance in the data which seems pretty good given the number of data points we have

Summary of the model

```r
summary(sharkGlm)
```

```
##
## Call:
## glm(formula = deal ~ ., family = "binomial", data = sharkSparseGlm)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6520  -0.9075   0.0043   0.8931   2.3322
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -0.17767    0.18969  -0.937  0.34896
## devic       -0.59607    0.54699  -1.090  0.27583
## new          0.69937    0.73582   0.950  0.34188
## retail       0.30726    0.85748   0.358  0.72010
## two          1.29668    0.82307   1.575  0.11516
## children     0.83308    0.60120   1.386  0.16584
## easi         2.27675    0.91416   2.491  0.01275 *
## make         0.46346    0.34950   1.326  0.18482
## one         -1.24307    0.60040  -2.070  0.03842 *
## play         0.26315    0.81827   0.322  0.74776
## provid       0.53599    0.78483   0.683  0.49465
## turn         1.11711    0.91453   1.222  0.22189
## organ        0.91355    0.80915   1.129  0.25889
## servic      -0.17389    0.52792  -0.329  0.74185
## women       -3.15274    1.26461  -2.493  0.01267 *
## offer        0.21128    0.80614   0.262  0.79325
## design       0.76810    0.38660   1.987  0.04694 *
## flavor       0.76317    0.61253   1.246  0.21279
## food        -0.07987    0.53389  -0.150  0.88108
## includ      -0.40064    0.76078  -0.527  0.59846
## line         0.93200    0.60501   1.540  0.12344
## product      0.27990    0.36383   0.769  0.44171
## sold        -0.53797    1.04513  -0.515  0.60674
## store       -0.05653    0.75156  -0.075  0.94004
## activ       -1.00909    0.87792  -1.149  0.25039
## apparel      1.58982    0.94642   1.680  0.09299 .
## brand       -1.92803    0.88518  -2.178  0.02940 *
## cloth        0.03739    0.52142   0.072  0.94284
## look        -0.39271    0.71208  -0.552  0.58129
## mix         -0.25894    0.57835  -0.448  0.65435
## attach       0.38079    0.74198   0.513  0.60781
## start        0.57069    1.14417   0.499  0.61794
## get          0.12632    0.81483   0.155  0.87680
## work         1.40915    0.72925   1.932  0.05332 .
## made        -1.41024    0.47280  -2.983  0.00286 **
## help        -0.23387    0.48594  -0.481  0.63032
## fit         -0.72387    0.60953  -1.188  0.23500
## combin      -1.75758    1.01827  -1.726  0.08434 .
## fun         -1.56313    0.75591  -2.068  0.03865 *
## keep         0.48888    0.67098   0.729  0.46624
## kid          0.10168    0.57107   0.178  0.85869
## compani      0.45641    0.46873   0.974  0.33020
## remov       -2.47430    1.02219  -2.421  0.01550 *
## also         1.16345    0.58856   1.977  0.04807 *
## babi        -0.94674    0.65322  -1.449  0.14724
## can          0.09051    0.37561   0.241  0.80958
## easier       2.00631    1.46507   1.369  0.17087
## like         0.01036    0.60064   0.017  0.98623
## protect      0.26619    0.65823   0.404  0.68592
## quick       -2.36049    1.15549  -2.043  0.04107 *
```

```
## size         2.94931    0.93419   3.157  0.00159 **
## solut        1.26399    1.05539   1.198  0.23105
## time         1.28841    0.74155   1.737  0.08231 .
## yet          0.20955    1.12089   0.187  0.85170
## use         -0.06683    0.39625  -0.169  0.86608
## back        -0.01360    0.78218  -0.017  0.98613
## buy         -0.57352    0.78602  -0.730  0.46560
## sell         0.91737    0.71757   1.278  0.20110
## year        -0.06352    1.14106  -0.056  0.95561
## accessori   -0.90700    0.71026  -1.277  0.20160
## color        0.69742    0.88990   0.784  0.43321
## onlin        0.16003    0.54731   0.292  0.76999
## user         0.32984    0.69634   0.474  0.63573
## bar         -0.82809    0.55195  -1.500  0.13354
## enjoy        0.04599    0.98137   0.047  0.96262
## ingredi      2.17027    1.00209   2.166  0.03033 *
## just         0.06628    0.58969   0.112  0.91050
## market       0.86970    0.96377   0.902  0.36684
## natur       -0.39297    0.61273  -0.641  0.52130
## safe        -1.30316    0.77327  -1.685  0.09194 .
## well         1.78603    1.14106   1.565  0.11753
## famili       0.58200    0.75647   0.769  0.44168
## fashion     -0.29053    0.71166  -0.408  0.68310
## custom      -1.91601    0.77278  -2.479  0.01316 *
## high         0.92217    0.78755   1.171  0.24162
## person      -0.45149    1.03496  -0.436  0.66267
## around      -1.02038    0.76923  -1.326  0.18468
## materi       1.08951    1.00965   1.079  0.28054
## featur      -1.73428    0.78422  -2.211  0.02700 *
## live         0.02478    0.87648   0.028  0.97745
## instead     -1.94475    1.09696  -1.773  0.07625 .
## uniqu        0.01687    0.76337   0.022  0.98237
## allow       -0.71602    0.60563  -1.182  0.23710
## special     -1.54994    1.01295  -1.530  0.12599
## dog          0.12051    0.62796   0.192  0.84782
## need         2.74190    1.05635   2.596  0.00944 **
## cover       -0.41931    0.58648  -0.715  0.47464
## allnatur    -0.76387    1.03740  -0.736  0.46153
## blend        0.96898    0.86536   1.120  0.26283
## patent       0.22001    0.81690   0.269  0.78768
## place       -0.05887    1.04808  -0.056  0.95521
## easili       0.43469    0.90385   0.481  0.63056
## way         -0.30611    0.72949  -0.420  0.67476
## busi        -0.84546    0.69708  -1.213  0.22519
## mobil        0.01509    0.80943   0.019  0.98512
## even         0.91622    0.68486   1.338  0.18095
## take        -0.57814    0.74355  -0.778  0.43684
## creat        0.66701    0.72152   0.924  0.35525
## usa          0.66137    1.17996   0.561  0.57513
## will        -1.57591    0.97676  -1.613  0.10666
```

```
## give         1.09284    0.86844   1.258  0.20825
## want        -0.92222    0.86997  -1.060  0.28912
## shoe         1.23385    0.64766   1.905  0.05677 .
## bag         -0.97544    0.92379  -1.056  0.29101
## money        0.59658    0.85698   0.696  0.48634
## power       -0.39019    0.72871  -0.535  0.59234
## produc      -0.09062    0.77561  -0.117  0.90699
## without      0.23174    0.62851   0.369  0.71234
## premium     -1.22986    1.01095  -1.217  0.22378
## fresh       -3.36004    1.28681  -2.611  0.00902 **
## bring       -2.57999    1.38571  -1.862  0.06262 .
## come        -0.82730    0.73867  -1.120  0.26272
## serv         0.30549    0.94272   0.324  0.74590
## avail       -1.52485    0.90081  -1.693  0.09050 .
## small       -0.06400    1.05668  -0.061  0.95171
## pad          1.08965    0.83104   1.311  0.18979
## system      -0.49703    0.62129  -0.800  0.42371
## home        -0.42010    0.65448  -0.642  0.52095
## tool        -1.62326    0.84051  -1.931  0.05345 .
## contain      0.16777    1.02857   0.163  0.87043
## water       -0.16442    0.53522  -0.307  0.75869
## peopl       -0.36533    0.73559  -0.497  0.61943
## day          1.69473    1.40867   1.203  0.22895
## packag       5.92664    2.47763   2.392  0.01675 *
## now         -0.64595    1.41525  -0.456  0.64809
## treat       -0.22294    0.82369  -0.271  0.78665
## everi       -2.06315    1.13288  -1.821  0.06858 .
## rang        -0.05531    1.14176  -0.048  0.96136
## drink       -0.05580    0.77962  -0.072  0.94295
## app          0.41453    0.70253   0.590  0.55515
## style        0.19912    0.88743   0.224  0.82246
## dont         4.32004    1.48333   2.912  0.00359 **
## simpl        0.44951    0.88535   0.508  0.61165
## great        4.74387    1.60567   2.954  0.00313 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 686.12  on 494  degrees of freedom
## Residual deviance: 503.62  on 361  degrees of freedom
## AIC: 771.62
##
## Number of Fisher Scoring iterations: 6
```

Top words contributing to othe classification

```
words <- as.data.frame((coef(summary(sharkGlm))[coef(summary(sharkGlm))[,4] <
0.05,3:4]))
```

```
colnames(words) <- c("pvalue", "zvalue")
words
```

```
##            pvalue      zvalue
## easi     2.490548 0.012754611
## one     -2.070389 0.038415971
## women   -2.493050 0.012665118
## design   1.986822 0.046942117
## brand   -2.178118 0.029397267
## made    -2.982741 0.002856798
## fun     -2.067878 0.038651491
## remov   -2.420584 0.015495580
## also     1.976782 0.048066275
## quick   -2.042843 0.041067982
## size     3.157079 0.001593581
## ingredi  2.165739 0.030331175
## custom  -2.479374 0.013161317
## featur  -2.211470 0.027003285
## need     2.595650 0.009441229
## fresh   -2.611144 0.009023997
## packag   2.392064 0.016753938
## dont     2.912398 0.003586653
## great    2.954445 0.003132317
```

Other than made which was significant predictor in other two mmodel all other words are different in a logistic regressions top 5 words. Here the top 5 words are

- Size, has a postive effect on the deal decision
- made, has a negative effect on the deal decision
- great, has a positive effect on the deal decision
- dont, has a positive effect on the deal decision
- fresh, has a negative effect on the deal decision
- need, has a positve effect on the deal decision

Accuracy of the model

```
accuracyGlm <- (sharkGlmCM[1]+sharkGlmCM[4])/sum(sharkGlmCM)
round(accuracyGlm * 100, 2)
```

```
## [1] 71.31
```

The accuracy is far better than CART at 71.31%

Improved Model

Now we will look at the same models after adding one more derived parameter ratio by using askedFor and valuation variables from the data set

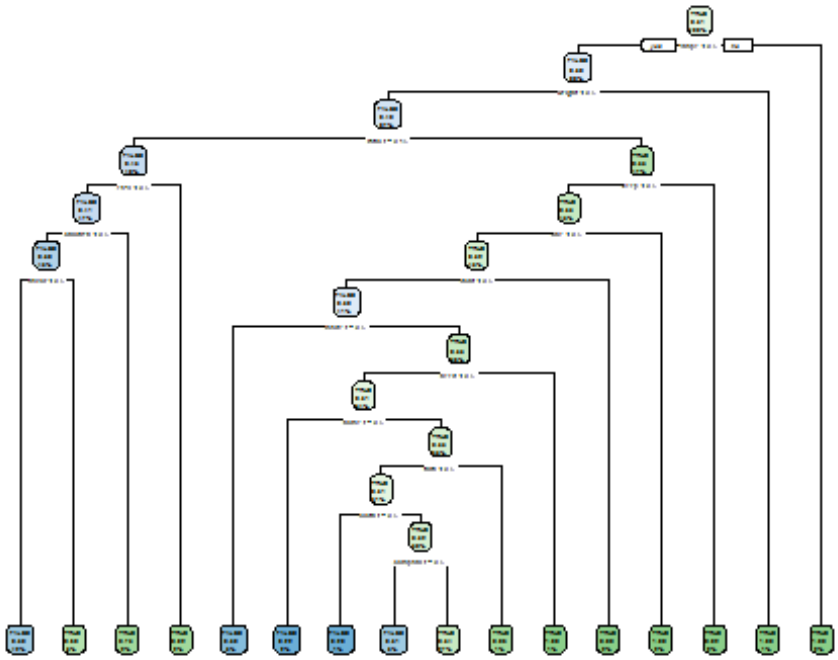
```
sharkSparse$ratio <- shark$askedFor/shark$valuation
```

Building the CART model

```
sharkCARTImp <- rpart(deal ~ ., data=sharkSparse, method="class")
```

Visualizing the model

```
rpart.plot(sharkCARTImp)
```



Ratio doesnot take part in the model, which signifies adding the ratio attribute does not change the model performance

COnfusion matrix

```
sharkCARTPredImp <- predict(sharkCARTImp, data=sharkSparse)
sharkCARTImpCM <- table("Actual" = sharkSparse$deal, "Prediction" = sharkCART
PredImp[,2] > 0.5)
sharkCARTImpCM
```

```
##        Prediction
## Actual  FALSE TRUE
##   FALSE   171   73
##   TRUE     95  156
```

Understanding the model accurady

```
accuracyImpCART <- (sharkCARTImpCM[1]+sharkCARTImpCM[4])/sum(sharkCARTImpCM)
round(accuracyImpCART * 100, 2)

## [1] 66.06
```

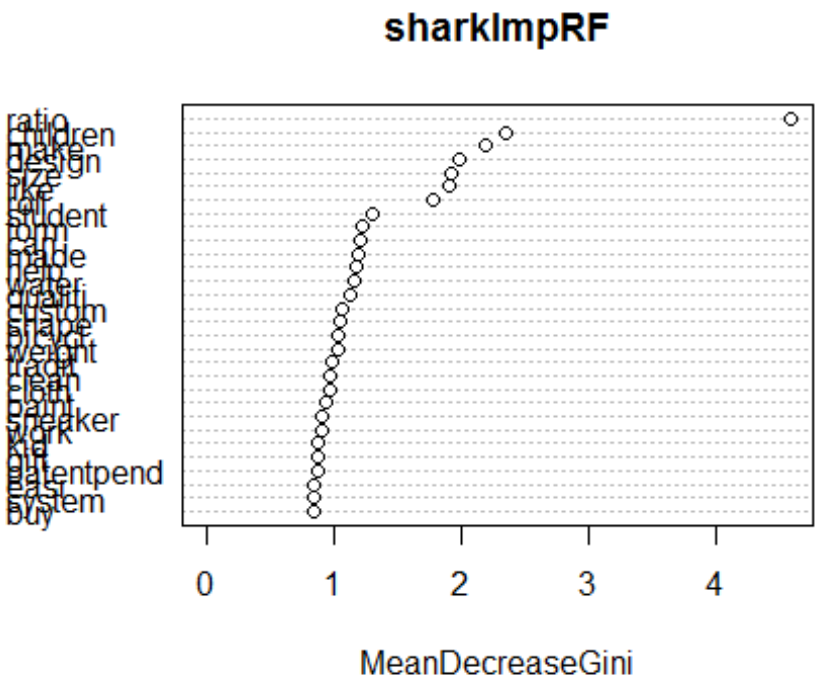
As expected the accurary of the model didnot change even a bit and stays at 65.66%

Building Random forest

```
sharkImpRF <- randomForest(deal~., data = sharkSparse, ntree=25)
sharkImpRFCM <- table("Actual" = sharkSparse$deal, "Prediction" = sharkImpRF$
predicted)
```

Visualizing the model

```
varImpPlot(sharkImpRF)
```



Top 5 words are

- easier
- make
- custom
- made
- water

Except for make and made other words have changed. But ratio still did not find a place in the top influencers

Confusion matrix

```
sharkImpRFCM
```

```
##        Prediction
## Actual  FALSE TRUE
##   FALSE   139  105
##   TRUE    125  126
```

Understandin the accuracy of the model

```
accuracyImpRF <- (sharkImpRFCM[1]+sharkImpRFCM[4])/sum(sharkImpRFCM)
round(accuracyImpRF * 100, 2)
```

```
## [1] 53.54
```

The model accuracy is 54.14% which is slightly better than 53.33%. But we are unable to infer the impact of ratio on the overall model.

Building the logistic regression model

Handling Sparce Matrix

```
sharkSparseGlm$ratio <- shark$askedFor/shark$valuation
```

Building the model

```
sharkImpGlm <- glm(deal~., data = sharkSparseGlm, family = "binomial")
sharkImpGlmCM <- table("Actual" = sharkSparseGlm$deal, "Prediction" = sharkIm
pGlm$fitted.values > 0.5)
```

Confusion Matrix

```
sharkImpGlmCM
```

```
##        Prediction
## Actual  FALSE TRUE
##   FALSE   178   66
##   TRUE     62  189
```

McFaden R2 to evaluate the model

```
round((1 - (sharkImpGlm$deviance/sharkImpGlm$null.deviance))*100, 2)
```

```
## [1] 27.28
```

The model explains about 27% of variance in the data which slightly better than the base model without ratio

Summary of the model

```
summary(sharkImpGlm)
```

```
##
## Call:
## glm(formula = deal ~ ., family = "binomial", data = sharkSparseGlm)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.75023  -0.90122   0.00225   0.85965   2.32663
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.2821288  0.2902739   0.972 0.331080
## devic       -0.5287771  0.5498209  -0.962 0.336187
## new          0.7346804  0.7429128   0.989 0.322703
## retail       0.1583240  0.8561028   0.185 0.853279
## two          1.2299505  0.8381021   1.468 0.142229
## children     0.8061535  0.5967511   1.351 0.176726
## easi         2.2819228  0.9288536   2.457 0.014022 *
## make         0.4884140  0.3538803   1.380 0.167535
## one         -1.2726933  0.6016710  -2.115 0.034407 *
## play         0.2375324  0.8162008   0.291 0.771035
## provid       0.4996417  0.7859688   0.636 0.524971
## turn         1.1092403  0.9341494   1.187 0.235057
## organ        0.8873340  0.8172862   1.086 0.277608
## servic      -0.1755188  0.5296340  -0.331 0.740345
## women       -2.9660547  1.2720971  -2.332 0.019720 *
## offer        0.1248215  0.8109918   0.154 0.877679
## design       0.7841282  0.3884347   2.019 0.043520 *
## flavor       0.7981068  0.6172024   1.293 0.195975
## food         0.0159485  0.5465754   0.029 0.976722
## includ      -0.4984689  0.7703127  -0.647 0.517568
## line         1.0578009  0.6061614   1.745 0.080971 .
## product      0.2453630  0.3668731   0.669 0.503626
```

```
## sold        -0.5387774  1.0213370  -0.528 0.597831
## store        0.0525159  0.7509123   0.070 0.944244
## activ       -1.0552975  0.8766283  -1.204 0.228661
## apparel      1.5186469  0.9557781   1.589 0.112080
## brand       -1.9573955  0.8855973  -2.210 0.027087 *
## cloth        0.0393248  0.5181956   0.076 0.939508
## look        -0.3231745  0.7111303  -0.454 0.649504
## mix         -0.2392137  0.5899832  -0.405 0.685141
## attach       0.5476183  0.7570954   0.723 0.469487
## start        0.4674196  1.1475950   0.407 0.683785
## get          0.1874267  0.8317057   0.225 0.821705
## work         1.5421598  0.7347252   2.099 0.035820 *
## made        -1.4400839  0.4758661  -3.026 0.002476 **
## help        -0.2272898  0.4892967  -0.465 0.642273
## fit         -0.7996054  0.6280635  -1.273 0.202973
## combin      -1.9605672  1.0451599  -1.876 0.060675 .
## fun         -1.5190860  0.7481814  -2.030 0.042319 *
## keep         0.6750069  0.6981254   0.967 0.333602
## kid          0.1393629  0.5721478   0.244 0.807557
## compani      0.4917168  0.4666098   1.054 0.291971
## remov       -2.4575400  1.0179602  -2.414 0.015771 *
## also         1.1747469  0.5921221   1.984 0.047260 *
## babi        -1.0282412  0.6603864  -1.557 0.119463
## can          0.0932682  0.3782445   0.247 0.805232
## easier       1.9917016  1.4314816   1.391 0.164117
## like        -0.0197595  0.6038258  -0.033 0.973895
## protect      0.2518353  0.6647769   0.379 0.704816
## quick       -2.3873920  1.1675911  -2.045 0.040883 *
## size         3.2471253  0.9677267   3.355 0.000792 ***
## solut        1.2492806  1.0674572   1.170 0.241867
## time         1.2264687  0.7476310   1.640 0.100907
## yet          0.1358276  1.1281940   0.120 0.904171
## use         -0.0779556  0.3992107  -0.195 0.845178
## back         0.0659766  0.7850840   0.084 0.933026
## buy         -0.6275613  0.7882030  -0.796 0.425920
## sell         0.8927255  0.7146816   1.249 0.211620
## year         0.1200661  1.1366847   0.106 0.915877
## accessori   -0.9691396  0.7253866  -1.336 0.181539
## color        0.6252176  0.8954745   0.698 0.485054
## onlin        0.1442828  0.5497074   0.262 0.792958
## user         0.3599231  0.6977177   0.516 0.605954
## bar         -0.7845730  0.5568331  -1.409 0.158838
## enjoy        0.0243666  0.9860689   0.025 0.980286
## ingredi      2.1513149  1.0215584   2.106 0.035212 *
## just         0.0234546  0.5917758   0.040 0.968385
## market       0.9481130  0.9639571   0.984 0.325330
## natur       -0.4389747  0.6172526  -0.711 0.476976
## safe        -1.3649858  0.7915894  -1.724 0.084643 .
## well         1.8760014  1.1721008   1.601 0.109477
## famili       0.6326974  0.7669025   0.825 0.409369
```

```
## fashion     -0.2214859  0.7140279  -0.310 0.756415
## custom      -1.9618191  0.7767474  -2.526 0.011547 *
## high         0.7504550  0.7937264   0.945 0.344412
## person      -0.3115964  1.0317567  -0.302 0.762648
## around      -0.9103654  0.7710410  -1.181 0.237723
## materi       0.8678900  1.0358954   0.838 0.402134
## featur      -1.6984612  0.7860249  -2.161 0.030709 *
## live        -0.0009388  0.8773897  -0.001 0.999146
## instead     -1.8578304  1.0880864  -1.707 0.087742 .
## uniqu       -0.0703153  0.7671342  -0.092 0.926968
## allow       -0.7736069  0.6097458  -1.269 0.204535
## special     -1.7278646  1.0159603  -1.701 0.088995 .
## dog          0.1008853  0.6212705   0.162 0.871002
## need         2.8485733  1.0618568   2.683 0.007304 **
## cover       -0.3954584  0.5861713  -0.675 0.499900
## allnatur    -0.8757022  1.0490735  -0.835 0.403865
## blend        1.0109272  0.8614523   1.174 0.240589
## patent       0.2595741  0.8286969   0.313 0.754105
## place       -0.0444010  1.0665936  -0.042 0.966795
## easili       0.4987472  0.9194096   0.542 0.587498
## way         -0.2952868  0.7361229  -0.401 0.688319
## busi        -0.8399069  0.6999917  -1.200 0.230185
## mobil        0.0773692  0.8068496   0.096 0.923608
## even         0.9161555  0.6850777   1.337 0.181124
## take        -0.6735818  0.7454477  -0.904 0.366211
## creat        0.8153396  0.7334053   1.112 0.266260
## usa          0.6594868  1.1989471   0.550 0.582282
## will        -1.5482601  0.9902546  -1.563 0.117936
## give         1.0191718  0.8674501   1.175 0.240033
## want        -0.9333418  0.8629040  -1.082 0.279418
## shoe         1.3077221  0.6465023   2.023 0.043097 *
## bag         -0.9373451  0.9283876  -1.010 0.312664
## money        0.5069691  0.8528906   0.594 0.552236
## power       -0.4628465  0.7316193  -0.633 0.526973
## produc      -0.1586829  0.7641278  -0.208 0.835490
## without      0.3089673  0.6354591   0.486 0.626817
## premium     -1.1563793  0.9942452  -1.163 0.244800
## fresh       -3.5426910  1.2970867  -2.731 0.006309 **
## bring       -2.6130288  1.4115870  -1.851 0.064151 .
## come        -0.8983178  0.7468274  -1.203 0.229036
## serv         0.4108024  0.9515592   0.432 0.665949
## avail       -1.5504598  0.9030247  -1.717 0.085986 .
## small       -0.0244662  1.0879694  -0.022 0.982059
## pad          1.1479988  0.8446813   1.359 0.174118
## system      -0.5817395  0.6262898  -0.929 0.352958
## home        -0.4333121  0.6665091  -0.650 0.515614
## tool        -1.7388228  0.8561628  -2.031 0.042260 *
## contain      0.2497143  1.0620346   0.235 0.814109
## water       -0.2393919  0.5448189  -0.439 0.660374
## peopl       -0.4289946  0.7491445  -0.573 0.566884
```

```
## day          1.5759538  1.4076072   1.120 0.262885
## packag       6.3160312  2.5507677   2.476 0.013282 *
## now         -0.5700840  1.3880974  -0.411 0.681297
## treat       -0.2317522  0.8228999  -0.282 0.778228
## everi       -2.0995762  1.1413169  -1.840 0.065826 .
## rang         0.2028573  1.1445398   0.177 0.859321
## drink        0.1229879  0.8104302   0.152 0.879379
## app          0.2572439  0.7120711   0.361 0.717904
## style        0.2410798  0.8866820   0.272 0.785707
## dont         4.3692197  1.4889017   2.935 0.003341 **
## simpl        0.4371247  0.8820252   0.496 0.620182
## great        4.6217376  1.6132535   2.865 0.004172 **
## ratio       -2.6794380  1.2928749  -2.072 0.038222 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 686.12  on 494  degrees of freedom
## Residual deviance: 498.97  on 360  degrees of freedom
## AIC: 768.97
##
## Number of Fisher Scoring iterations: 6
```

Top words contributing to othe classification

```
predictors <- as.data.frame((coef(summary(sharkImpGlm))[coef(summary(sharkImp
Glm))[,4] < 0.05,3:4]))
colnames(predictors) <- c("pvalue", "zvalue")
predictors
```

```
##            pvalue       zvalue
## easi     2.456709 0.0140216319
## one     -2.115265 0.0344074052
## women   -2.331626 0.0197203703
## design   2.018687 0.0435197200
## brand   -2.210255 0.0270874937
## work     2.098961 0.0358203001
## made    -3.026238 0.0024761767
## fun     -2.030371 0.0423188286
## remov   -2.414181 0.0157706292
## also     1.983961 0.0472602271
## quick   -2.044716 0.0408828848
## size     3.355416 0.0007924584
## ingredi  2.105915 0.0352117611
## custom  -2.525685 0.0115473025
## featur  -2.160824 0.0307089578
## need     2.682634 0.0073044835
## shoe     2.022765 0.0430973971
## fresh   -2.731268 0.0063091207
```

```
## tool    -2.030949 0.0422601885
## packag   2.476130 0.0132815386
## dont     2.934525 0.0033405837
## great    2.864855 0.0041719996
## ratio   -2.072465 0.0382220967
```

There is no change in the top words even in logistics regression, but ratio has a statistically significant effect on the model with a pvalue of 0.038 which is less than 0.05.

Accuracy of the model

```
accuracyImpGlm <- (sharkImpGlmCM[1]+sharkImpGlmCM[4])/sum(sharkImpGlmCM)
round(accuracyImpGlm * 100, 2)

## [1] 74.14

round(c("CART" = accuracyCART, "RF" = accuracyRF, "GLM" = accuracyGlm) * 100,
2)

##  CART    RF   GLM
## 65.66 54.95 71.31

round(c("CART" = accuracyImpCART, "RF" = accuracyImpRF, "GLM" = accuracyImpGl
m) * 100, 2)

##  CART    RF   GLM
## 66.06 53.54 74.14
```

Overall inference from various models

The description of the submission can be used to predict if a submission will make the deal or not. All three models suggest that the top words are “make”, “made”. While logistic regression suggest the word “size” has more significant effect on the judges to make a deal for a submission.

Ratio, which is a function of value asked for and the expert valuation of the submission, has no effect when it comes to CART and has a slight unexplainable effect in RF.

But Logistics regression explains the ralationship well between the deal outcome and ratio. This improved the effectiveness of the model by 3%.

We could not use more “words” as the data points are far less than the words extracted out of the description. If we had more data points, we could have used more words instead of just 131 words to improve the effectiveness of logistic regression even further.